NPS55-86-025

NAVAL POSTGRADUATE SCHOOL

Monterey, California

TECHNICAL

GRAPHICAL ANALYSIS OF SOME PSEUDO-RANDOM NUMBER GENERATORS

PETER A. W. LEWIS

DECEMBER 1986

Prepared for:
Naval Postgraduate School
Monterey, CA 93943-5000

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral R. C. Austin
Superintendent

D. A. Schrady
Provost

This report was prepared by:

UNCLASSIFIED

URITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1 REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2 SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2 DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS55-86-025	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6 NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) Code 55	7a NAME OF MONITORING ORGANIZATION
6 ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code)
NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

. TITLE (Include Security Classification)

GRAPHICAL ANALYSIS OF SOME PSEUDO-RANDOM NUMBER GENERATORS

. PERSONAL AUTHOR(S)

Lewis, Peter A. W.

3a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical	FROM ____ TO ____	December 1986	17

5. SUPPLEMENTARY NOTATION

7. COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

9. ABSTRACT (Continue on reverse if necessary and identify by block number)

There exist today many 'good' pseudo-random number generators; the problem is to retrieve them. We discuss three commonly used pseudo-random number generators, the first being RANDU, a notoriously bad generator, but one which is still occasionally used. The next is the widely used prime modulus, multiplicative congruential generator used in LL-RANDOMII, the Naval Postgraduate School random number package, and the last is the random number generator provided for microcomputers with the DOS operating system. This latter pseudo-random number generator is completely defective. Simple graphical methods for initial screening of pseudo-random number generators are given, and the problems which arise with bad pseudo-random number generators are detailed with graphics. Finally, recent work on obtaining even better pseudo-random number generators is discussed.

0 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
2a. NAME OF RESPONSIBLE INDIVIDUAL Peter A. W. Lewis	22b TELEPHONE (Include Area Code) (408)646-2283 22c OFFICE SYMBOL Code 55Lw

D FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office 1986—606-243

Errata
for
'Graphical Analysis of Some Pseudo-Random Number Generators'
by
Peter A. W. Lewis

Line 3 from the top of page 13 should read

'...addition of independent Uniform (0,1) random variables,
modulo 1 , gives ...'

Lines 7 and 8 from the top on page 3 should be replaced by the following sentence.

'Unfortunately, although this generator is highly defective, as we shall see in the next Section, it is still widely used. It is, for example, the generator supplied with the VMS operating system for DEC VAX machines.'

Graphical Analysis of some Pseudo-Random Number Generators.

Peter A. W. Lewis

Department of Operations Research
Naval Postgraduate School
Monterey, CA 93940.

Abstract

There exist today many 'good' pseudo-random number generators; the problem is to retrieve them. We discuss three commonly used pseudo-random number generators, the first being RANDU, a notoriously bad generator, but one which is still occasionally used. The next is the widely used prime modulus, multiplicative congruential generator used in LLRANDOMII, the Naval Postgraduate School random number package, and the last is the random number generator provided for microcomputers with the DOS operating system. This latter pseudo-random number generator is completely defective. Simple graphical methods for initial screening of pseudo-random number generators are given, and the problems which arise with bad pseudo-random number generators are detailed with graphics. Finally, recent work on obtaining even better pseudo-random number generators is discussed.

1 Introduction.

Pseudo-random number generators are required in a large variety of computational situations, particularly those of statistical and sytems simulations. The requirement that one have a 'good' generator almost goes without saying, but it is not always an easily obtained objective. The problem on the grossest level is that all pseudo-random number generators are deterministic sequences of numbers, and this seems to be at complete variance with the requirement that the pseudo-random number generator is required to simulate a sequence of independent and uniformly distributed random variables.

This conundrum will not be of concern here, although it has caused much soul searching; see, for example, the discussion in Knuth (1981, pp. 142-166). In simulations, the deterministic nature of pseudo-random number generators is actually a plus, because the simulation can be repeated or the random numbers can be used in a slightly different run of the simulation for the purposes of, say, variance reduction.

One general property of pseudo-random number generators is that they should have a long *cycle length*. This means that if we start the pseudo-random

number generator with a given number, called the *seed*, that number should not recur until a large number of other numbers have been generated. Moreover, this cycle length should not depend on which of the possible numbers we use as the starting value or *seed*. In the next Section, we will discuss linear congruential generators, and for these the maximum (full) cycle is the largest number which can be generated in the computer under consideration. For example, for a 31-bit machine the maximum cycle length obtainable is 2^{31}.

Note, too, that pseudo-random number generators actually generate integer numbers; these are converted to real numbers — Uniform (0,1) deviates — by dividing by the largest number that can be generated. A full cycle helps to make this uniform deviate as dense on the real line as possible. This is not the same as the uniform deviate having a uniform distribution on the unit interval. For that, any sequence of numbers generated by the generator must be uniformly distributed over (0,1).

Some specific generators will be discussed next; one which is known to be bad, one which is known to be good and one which is new and whose properties are undocumented. Then graphical tests of these pseudo-random number generators are presented. Following this, an illustration is given of the effects of using a bad generator. Finally, we discuss recent results on testing and documenting the properties of pseudo-random number generators, and we also discuss other schemes for random number generation which have been proposed.

2 Congruential Pseudo-Random Number Generators.

We now discuss a special and widely used type of pseudo-random number generator called a linear congruential generator. The idea is due to Lehmer (1951). Let U_i be the sequence of integers generated, where $i = 1, 2, \cdots$, and U_0 is the starting value or *seed*. Then

$$U_i = (aU_{i-1} + c) \bmod m. \tag{1}$$

Here a, c and m are integer constants, a being the *multiplier*, c being the *additive constant* and m being the *modulus* .

As remarked before, one wants a modulus which is as large as possible for a given machine. In addition, if m is a power of 2, then the division in the modulus operation becomes a shift operation, which is fast to implement in a digital computer. However, other considerations may make other choices of modulus preferable. As for the numbers a and c, their choice is crucial to the adequate behaviour of the pseudo-random number generator.

2.1 RANDU — a bad pseudo-random number generator.

As an example of a bad linear congruential generator, consider the case where $m = 2^{31}$ and

$$a = 2^{31} + 3,$$

which in binary notation is 1000000000000000011. This is the generator RANDU which was distributed with the IBM Scientific Subroutine Package. Actually, there was a percursor for 35-bit machines, but we will only discuss this 31-bit version. The choice of the constants was done mainly on the basis of computational considerations. Thus, since the multiplier has only three one-bits, the multiplication is extremely fast.

Unfortunately this generator, although widely used, is also highly defective, as we will see in the next Section.

2.2 Prime modulus multiplicative congruential generator.

If the constant c is chosen to be zero, one has a multiplicative congruential generator; clearly one must avoid zero and the maximum cycle length is just m-1. If m is a prime number, a choice which is not optimal from a computational view, and a is chosen to be a *positive primitive root* of m, then one has a *prime modulus multiplicative congruential generator*. The fact that a is a positive primitive root means that the generator cycles through all m-1 values before recycling, so that one has a linear congruential generator of maximum cycle length.

Now positive primitive roots may not exist for a given m, but fortunately number theorists have long been interested in this existence question and, for many primes, the roots are tabulated. In fact, for the number $2^{31} - 1$, which is the largest prime less than 2^{31} and thus ideal for use in a 31-bit machine, the numbers $a = 7$ and $a = 7^5$ are positive primitive roots.

The prime modulus multiplicative congruential generator with $m = 2^{31} - 1$ and $a = 7^5$ was developed by Lewis, Goodman and Miller (1969) and is widely used. It is the generator in APL and in the IMSL subroutine GGUBS, and is implemented in a very fast Assembly Language version in a package called LLRANDOMII at the Naval Postgraduate School (Lewis and Uribe, 1981). Incidental to this Assembly Language implementation is the fact that there is a division simulation algorithm, also due to Lehmer, which speeds up the mod operation. A FORTRAN Version for microcomputers is available in the Advanced Simulation and Statistics Package (Lewis, Orav and Uribe, 1986).

This LLRANDOMII generator was developed on the basis of extensive testing and has stood up well with time. Better prime modulus multiplicative congruential generators are now known; these will be discussed in Section 5.

2.3 The BASIC pseudo-random number generator for microcomputers.

The advent of microcomputers has opened new questions about random number generators. In particular, where good pseudo-random number generators are known to exist for large computers, how about 8-bit and 16-bit microcomputers?

We will be concerned here with the pseudo-random number generator which is provided with BASIC on microcomputers which use Microsoft DOS. What kind of generator it is, is unknown, since it is a trade secret, but it is presumably a linear congruential generator. In the next Section we will examine this

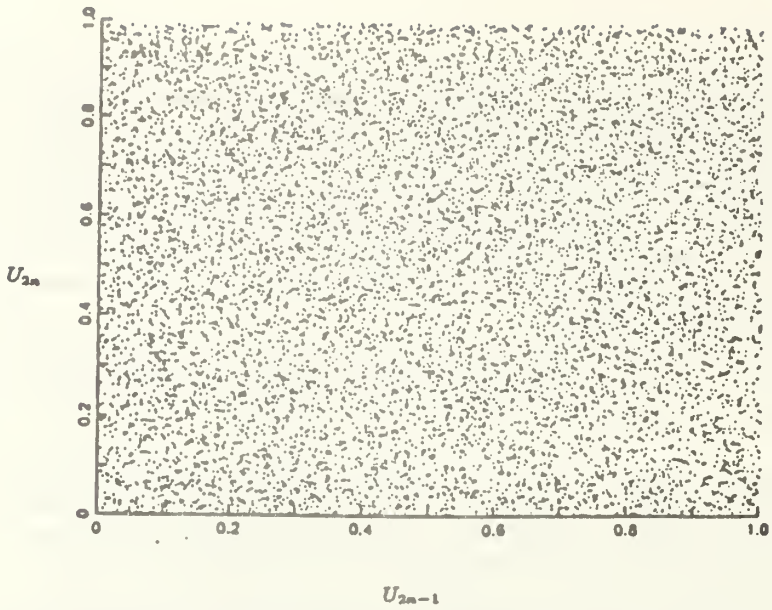

Figure 1: Two-dimensional scatter plot of sequential non-overlapping pairs of points from the pseudo-random number generator RANDU.

pseudo-random number generator, and demonstrate why one should always use a known pseudo-random number generator with documented properties; this BASIC pseudo-random number generator is, in fact, terrible.

3 Some Graphical tests of pseudo-random number generators.

3.1 Testing in general.

Testing of pseudo-random number generators is a time consuming and difficult task. However, it is possible to screen pseudo-random number generators on the basis of graphical tests which readily reveal gross flaws in the generator.

3.2 Two-dimensional scatter plots.

Clearly the random numbers must be uniformly distributed over their range and we will assume that this has been checked, at least graphically, and proceed to two-dimensional scatter plots of U_n *vs.* U_{n+1} for $n = 1, 2, \ldots N$. This plot tests for uniformity of the bivariate distribution of the pairs in the plane and for pairwise dependence. These tests can be formalized by counting the points which fall in the cells of a grid laid over the plot, but we do not discuss these tests here. In Figures 1, 2 and 3 these plots are shown for RANDU, LLRANDOMII and the BASIC generator, respectively. Actually, plots of non-overlapping pairs U_{2n} *vs.* U_{2n-1} are shown, for $n = 1, 2, \ldots, 10,000$.

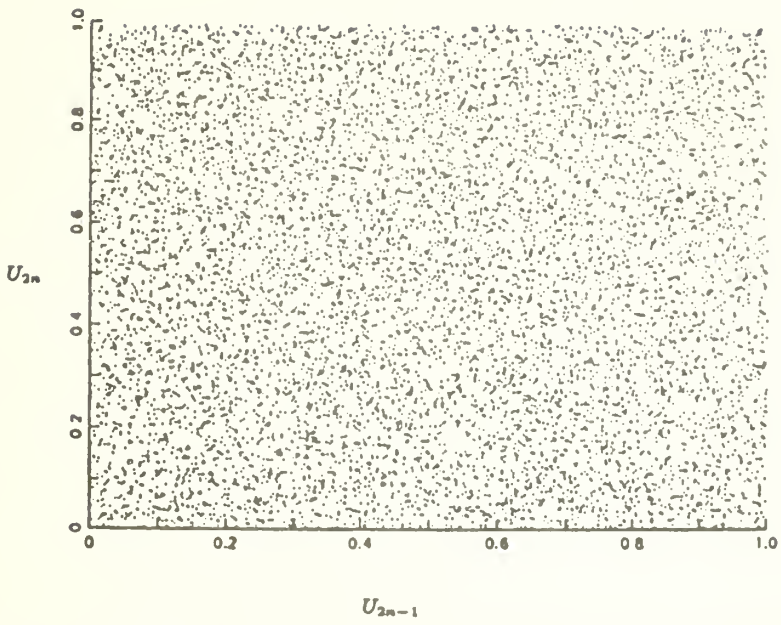

Figure 2: Two-dimensional scatter plot of sequential non-overlapping pairs of points from the pseudo-random number generator LLRANDOMII.

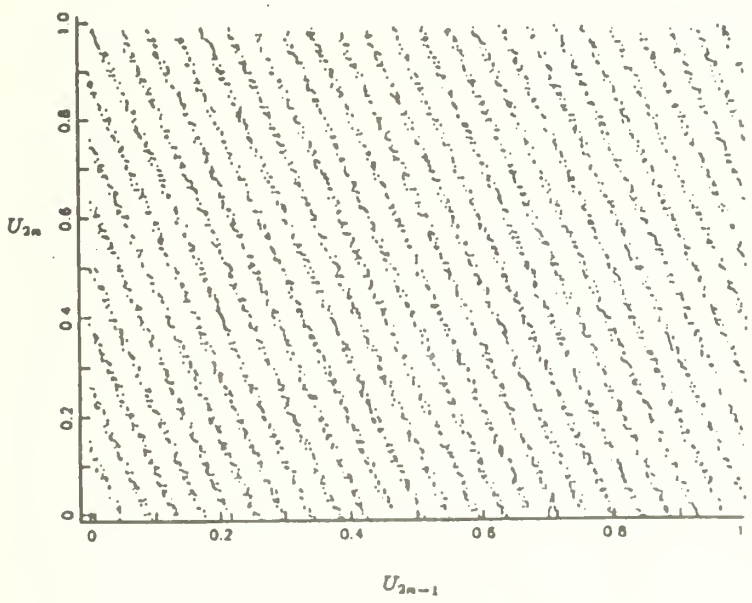

Figure 3: Two-dimensional scatter plot of sequential non-overlapping pairs of points from the pseudo-random number generator from Microsoft BASIC.

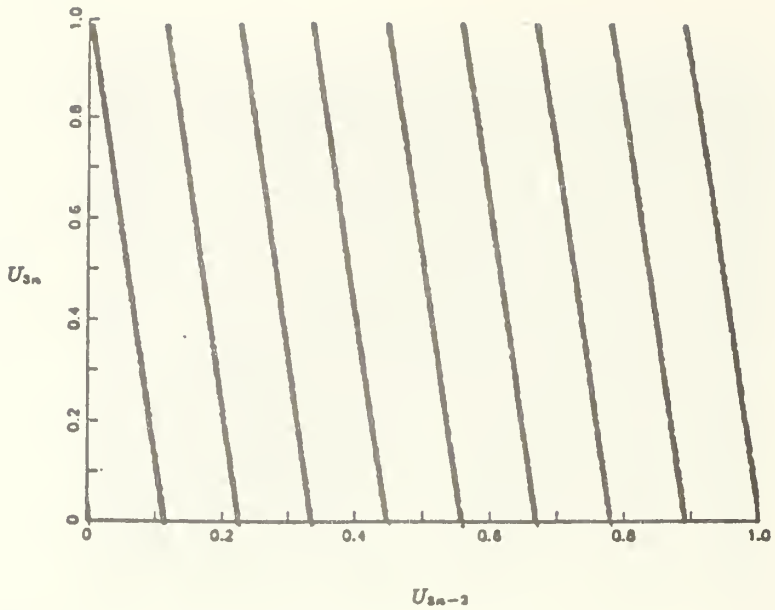

Figure 4: Two-dimensional slice in three-dimensional scatter plot of sequential non-overlapping triples of points from the pseudo-random number generator RANDU.

Note that for RANDU and LLRANDOMII there is no apparent deviation from randomness in the plane. However, for the BASIC pseudo-random number generator, there is definitely a striation in the scatter of points. One caveat must be mentioned here. These plots were done on a very high-resolution Tektronix 618 monitor driven by the IBM APL-based graphics package GRAFSTAT. On the monitor of the IBM PC, this pattern or striation is barely noticeable. However one of the students at the Naval Postgraduate School was astute enough to notice the possibility of a pattern and to investigate it further.

It is clear from this simplest of graphical tests that the BASIC generator is grossly defective, yet no one seems to have noticed or documented this fact.

3.3 Two-dimensional slices of three-dimensional scatter plots.

Obviously testing of pseudo-random number generators in two-dimensions is not enough, and one wants to proceed to higher dimensions. This is, however, difficult on a two-dimensional screen, although programs for rotation of three-dimensional point clouds do exist. A simple solution is to look at scatter plots of two-dimensional slices of points in a cube. Thus one plots U_{3n} *vs.* U_{3n-2} for triples $\{U_{3n}, U_{3n-1}, U_{3n-2}\}$ in which, say, $0.5 \leq U_{3n-1} < 0.51$. The drawback is that if one plots, say, 10,000 of these points, approximately 1,000,000 pairs will have to be generated. Clearly one might also want to look at other slices.

The result of these graphics is that one sees, first of all, just how truly bad the BASIC generator is (Figure 6), and also the fact that the generator RANDU is completely flawed (Figure 4). This has been shown analytically — see, for

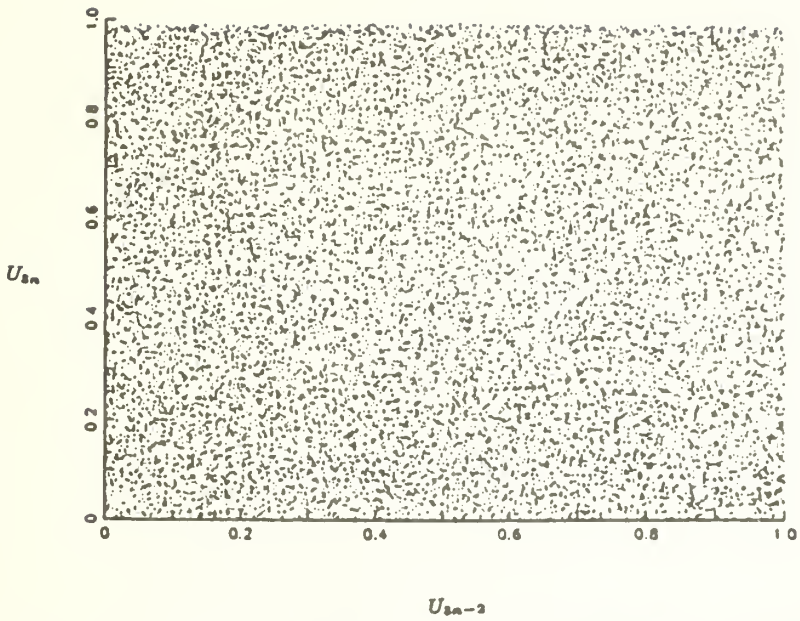

Figure 5: Two-dimensional slice in three-dimensional scatter plot of sequential non-overlapping triples of points from the pseudo-random number generator LLRANDOMII.

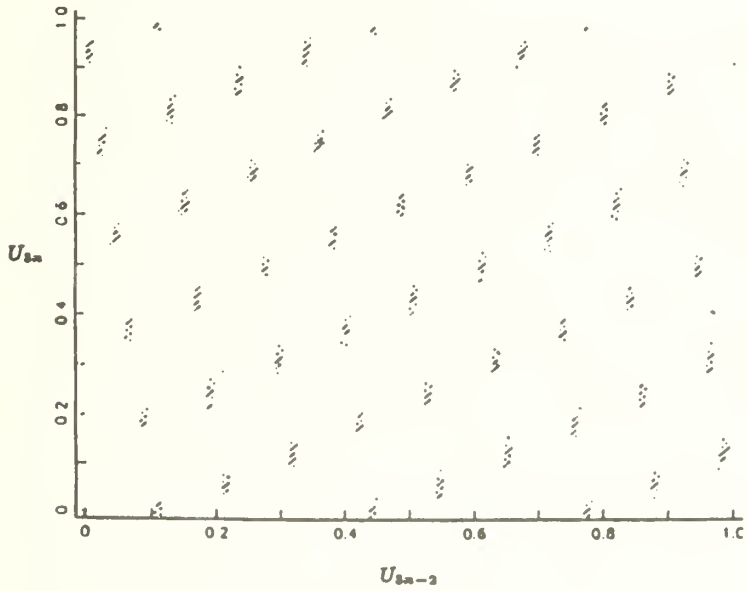

Figure 6: Two-dimensional slice in three-dimensional scatter plot of sequential non-overlapping triples of points from pseudo-random number generator from Microsoft BASIC.

example, Fishman (1978) — but was known to astrophysicists who regularly tried to use RANDU in the 1960's to produce random scatters of points in three-space. Nevertheless the generator was used on into the 1970's!

No obvious departure from randomness can be seen in Figure 5 for LLRANDOMII.

3.4 Higher-dimensional slicing.

Slices in higher dimensional scatter plots can be investigated and have been done for the pseudo-random number generator LLRANDOMII up to the sixth dimension. No departures from randomness are evident. More formal tests of pseudo-random number generators like LLRANDOMII will be discussed in Section 5.

4 What difference does it all make?

This question was posed to me by a student after I had showed him what I considered to be proof positive of the defects of pseudo-random number generators. It is a legitimate question, and the following example is the simplest illustrative but substantial case I have been able to come up with.

Many of the problems with pseudo-random number generators come from the use of recycling of pseudo-random numbers. It is a commonly used technique for speeding up simulations and is based on the following Theorem.

Theorem 1 *Let U be uniformly distributed on (0,1). Then conditional on the event $U \leq p$, the variable $U' = U/p$ is Uniform (0,1) and conditionally independent of U. Otherwise, conditional on the event $U > p$, the variable $U' = (1-U)/(1-p)$ is Uniform (0,1) and conditionally independent of U.*

What is actually happening with this trick of recycling is that low-order bits of the random number generator are being used and, since most testing of pseudo-random number generators is really concerned with the high-order bits, there is no guarantee that the theoretical properties of Uniform (0,1) random variables given in this theorem will hold for pseudo-random number generators.

To test this and show its effect on non-uniform random number generation, in the following example we use pseudo-random numbers which have *all* been stripped a given number of times by comparing them to $p = 1/2$.

The example is the generation of independent pairs of independent Normal (0,1) random variables by the modified Box-Muller or polar method. This method, with a modification due to von-Neumann for speed, uses triples of pseudo-random numbers, and thus should be fairly critically dependent on the 3-dimensional properties of the pseudo-random number generator. It is described in most books on simulation, and in particular in Fishman (1978, p. 412), so that it will not be repeated here.

Figure 7 shows a scatter plot of pairs of Normal random deviates generated by the modified Box-Muller method using the pseudo-random number generator LLRANDOMII with *no stripping*. To the eye, there is no obvious departure from randomness, i.e., no pattern, although the density of the points tapers off as one

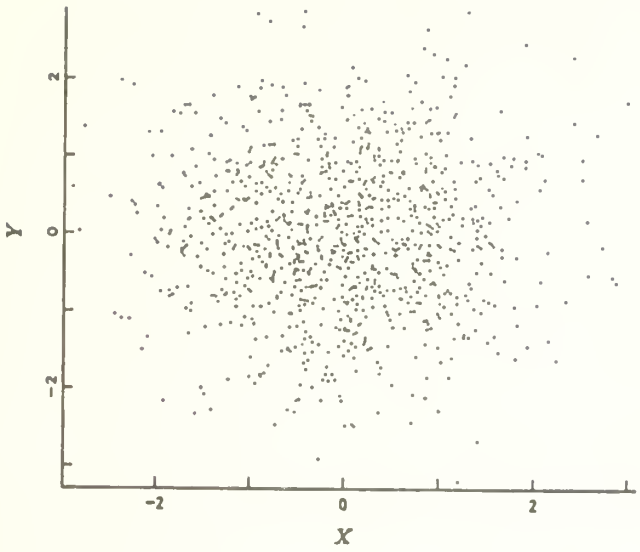

Figure 7: Plot of pairs of independent Normal random variables generated by the modified Box-Muller method using the pseudo-random number generator LLRANDOMII with *no stripping*.

moves from the center because of the bivariate Normal distribution of the points. In Figure 8 there is the same plot, but this time the pseudo-random number generator RANDU has been used. There is some intimation of a striation in the bottom of the plot, to the left of center, and in the top to the right of center. The plot is repeated in Figure 9 with the difference that the pseudo-random numbers from RANDU have been stripped of six bits. The striation seems more pronounced. In Figures 10 and 11, ten and sixteen bits, respectively, have been stripped and there is now no doubt that there is a pattern. Thus, the generated pairs are clearly not simulating independent Normal pairs.

What happens when LLRANDOMII is used with sixteen bits stripped? Figure 12 shows that there is no visible effect on the independence of the Normal pairs when the pseudo-random number generator is stripped. This is encouraging for the user of pseudo-random number generators and illustrates why this pseudo-random number generator has been widely used.

One other point is of interest here. The 'badness' which shows up in the joint structure of the simulated Normal pairs is not necessarily reflected in the marginal structure of the Normal pairs. Thus, in Figure 13 we give a Normal probability plot of the first variable of the pairs which were plotted in Figure 10. The plot is reasonably straight and, in fact, within the Kolmogorov-Smirnov bounds which are shown on the plot. With stripping of sixteen bits — corresponding to Figure 11 — the Normal probability plot shows that there is no longer marginal Normality in the simulated Normal pairs. But the discrepancy in the departure in the marginal distribution is in no sense as gross as the departure in the joint distribution.

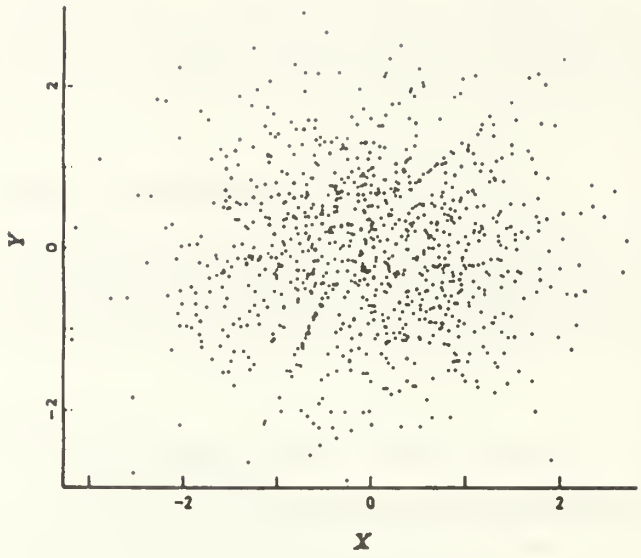

Figure 8: Plot of pairs of independent Normal random variables generated by the modified Box-Muller method using the pseudo-random number generator RANDU with *no stripping*.

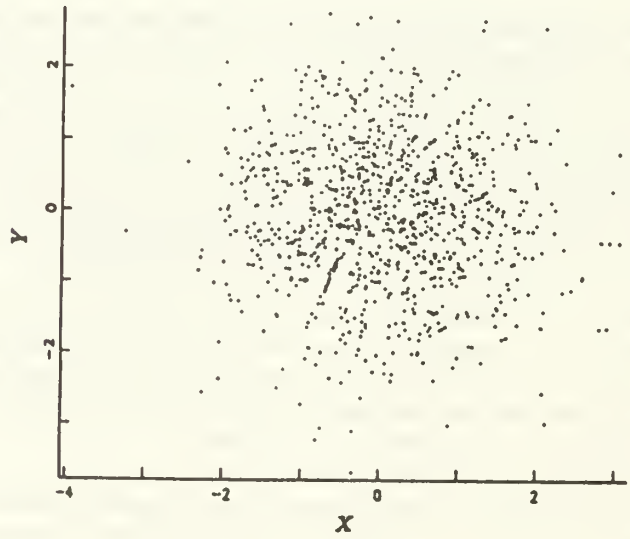

Figure 9: Plot of pairs of independent Normal random variables generated by the modified Box-Muller method using the pseudo-random number generator RANDU with *6 bits strippped*.

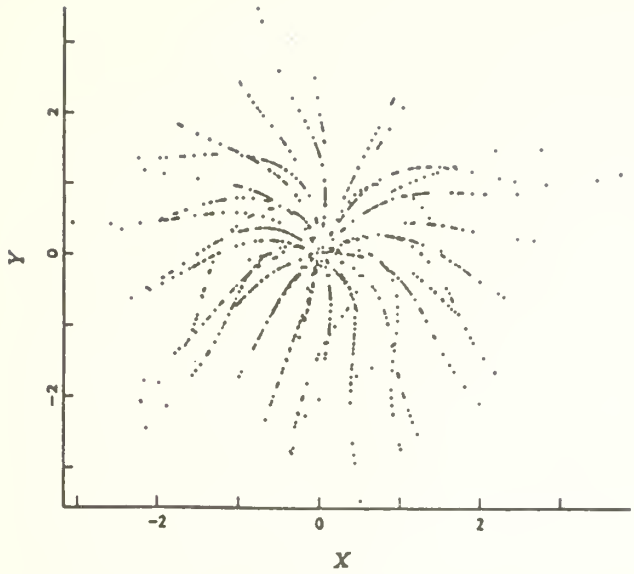

Figure 10: Plot of pairs of independent Normal random variables generated by the modified Box-Muller method using the pseudo-random number generator RANDU with *10 bits stripped.*

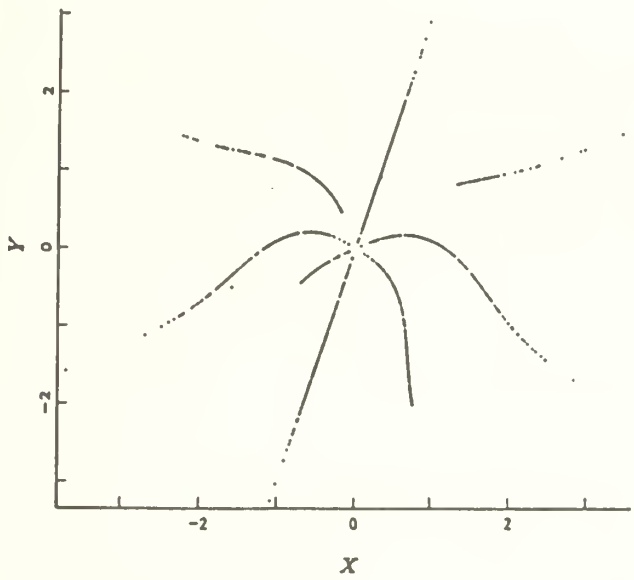

Figure 11: Plot of pairs of independent Normal random variables generated by the modified Box-Muller method using the pseudo-random number generator RANDU with *16 bits stripped.*

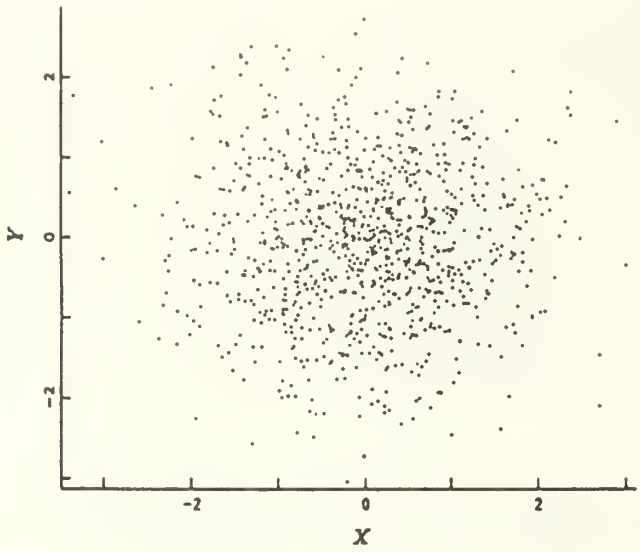

Figure 12: Plot of pairs of independent Normal random variables generated by the modified Box-Muller method using the pseudo-random number generator LLRANDOMII with *16 bits stripped.*

5 Further developments and references.

The development of pseudo-random number generators continues, despite the fact that generators such as that referred to as LLRANDOMII — a prime modulus multiplicative congruential generator with multiplier $a = 7^5$ and modulus $m = 2^{31} - 1$ — appear to be relatively good. In fact, the LLRANDOMII pseudo-random number generator is known to be defective, in a certain technical sense, in higher dimensions. We discuss here very briefly some of these recent developments in pseudo-random number generators.

5.1 Other multipliers for prime modulus multiplicative congruential generators.

In two massive studies, Fishman and Moore (see Fishman and Moore, 1985) investigated the properties of prime modulus multiplicative congruential generators like LLRANDOMII, but with other multipliers a. The result of this study was a list of preferred multipliers, of which 950706376 was ranked first, which give 'better' pseudo-random number generators than LLRANDOMII. The improvements obtained with these multipliers are too subtle to be seen with the simple graphical methods given in this paper. Several of these multipliers are available in, for example, the IMSL Statistics Library and in the Naval Postgraduate School Random Number Generator Package – LLRANDOMII.

5.2 The Wichman-Hill generator.

A very good, well tested, and unusual pseudo-random number generator for computers with only 16 bit arithmetic was given by Wichman and Hill (1982). It uses the fact that addition of independent Uniform (0,1) random variables gives another Uniform (0,1) random variable. Combination in this way of three prime modulus multiplicative congruential generators with small moduli gives a pseudo-random number generator with a very long cycle length. This combination scheme is, in fact, a form of *shuffling*; several of these schemes for improving the performance of pseudo-random number generators are known. A two-dimensional scatter plot of pairs from the Wichman-Hill generator is given in Figure 15, and the two-dimensional slice of a three-dimensional scatter plot is given in Figure 16. No obvious departures from randomness appear in the Figures. Tests of Normal pairs obtained with this generator also give good results, although one would not strip off too many bits in a generator which has essentially only 15 bits to start with.

5.3 Fibonacci generators.

Marsaglia has advocated a different type of pseudo-random number generator called a Fibonacci generator. For a discussion of these generators, and a summary of other recent developments on pseudo-random number generators, the reader is referred to the recent review article by Marsaglia (1984).

6 Conclusions: testing pseudo-random number generators.

We have shown that one can find very good pseudo-random number generators but also that the pseudo-random number generators provided in some packages and with some computer systems are defective. This leads to the following guidelines on the testing of pseudo-random number generators, which, when done completely, is a time consuming and arduous task, not to be undertaken lightly.

1. Testing is unnecessary in the sense that *very good* pseudo-random number generators are available.

2. Testing is necessary in the sense that bad pseudo-random number generators exist on many computer systems and are commonly used.

3. It is preferable to substitute a good pseudo-random number generator with documented properties for a pseudo-random number generator you do not know anything about.

And even if you follow the last of these rules, it is wise to use some of the graphical testing methods given in this paper to make sure that the implementation is correct.

7 Acknowledgements.

I am indebted to Professor E. J. Orav for comments on this paper and for the stimulation which led to its writing. Several students at the Naval Postgraduate School also contributed to its contents. This paper was prepared on a Compaq Portable 286 Computer using the version of LaTeX distributed by Personal TeX. The graphics was performed with the APL GRAFSTAT program which is available at the Naval Postgraduate School under a test site agreement with IBM Research. We are indebted to Dr. P.D. Welch for making the GRAFSTAt program available to us. Finally this research was supported by the Office of Naval Research under Grant RR014-05-01.

8 References.

Fishman, G.S. (1978). *Principles of Discrete Event Simulation.* Wiley: New York.

Fishman, G.S. and Moore, L. (1985). An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus $2^{31}-1$. *SIAM Journal of Scientific and Statistical Computing.*

Knuth, D.E. (1981). *The Art of Computer Programming. Volume 2. Seminumerical Algorithms, 2nd ed..* Addison-Wesley: Reading, Mass.

Lehmer, D.H. (1951). Mathematical Methods in Large-scale Computing Units. *Annals of the Computing Laboratory [Harvard University] , 26,* 141-146.

Lewis, P.A.W., Goodman, A.S., and Miller, J.M. (1969). A Pseudo-random Number Generator for the System 360. *IBM Systems Journal, 8,* 136-146.

Lewis, P.A.W. and Uribe, L.C. (1981). The New Naval Postgraduate School Random Number Generator Package LLRANDOMII. *Naval Postgraduate School Technical Report, PS-55-81-005.*

Lewis, P.A.W., Orav, E.J., and Uribe, L.C. (1986). *Advanced Simulation and Statistics Package.* Wadsworth & Brooks/Cole: Monterey, Ca.

Marsaglia, G. (1985). A Current View of Random Number Generators. *Computer Science and Statistics: Proceedings of the Sixteenth Symposium on the Interface.* L. Billard (ed.), North-Holland: Amsterdam, 3-10.

Wichman, B.A. and Hill, I.D. (1982). An Efficient and Portable Pseudo-Random Number Generator. *Applied Statistics, 31, 2,* 188-190.

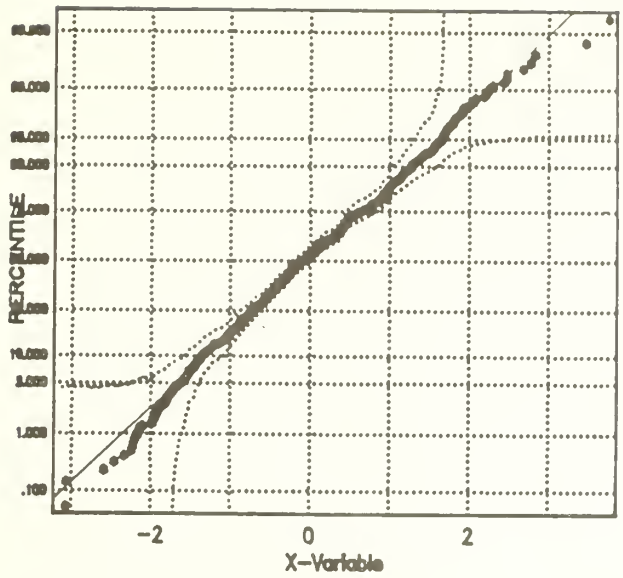

Figure 13: Normal probability plot for the first variable of the simulated Normal pairs whose scatter plot is given in Figure 10. This is the case of RANDU with *10 bits stripped.*

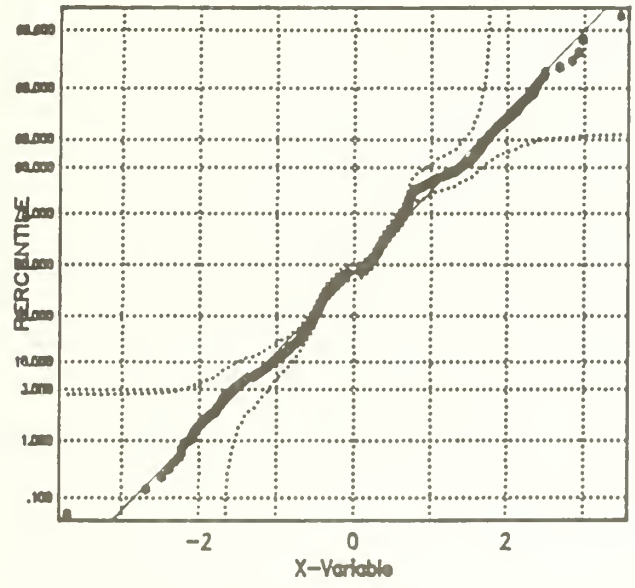

Figure 14: Normal probability plot for the first variable of the simulated Normal pairs whose scatter plot is given in Figure 11. This is the case of RANDU with *16 bits stripped.*

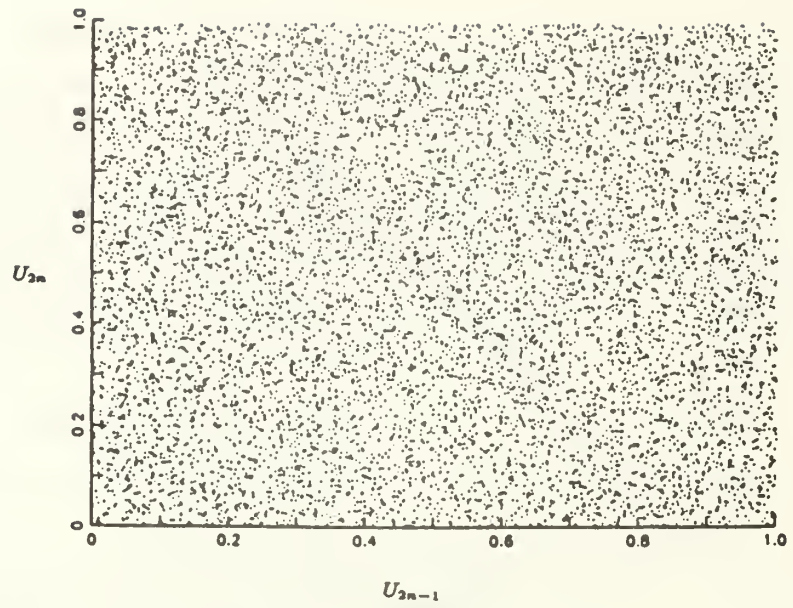

Figure 15: Two-dimensional scatter plot of sequential non-overlapping pairs of points from the Wichman-Hill pseudo-random number generator.

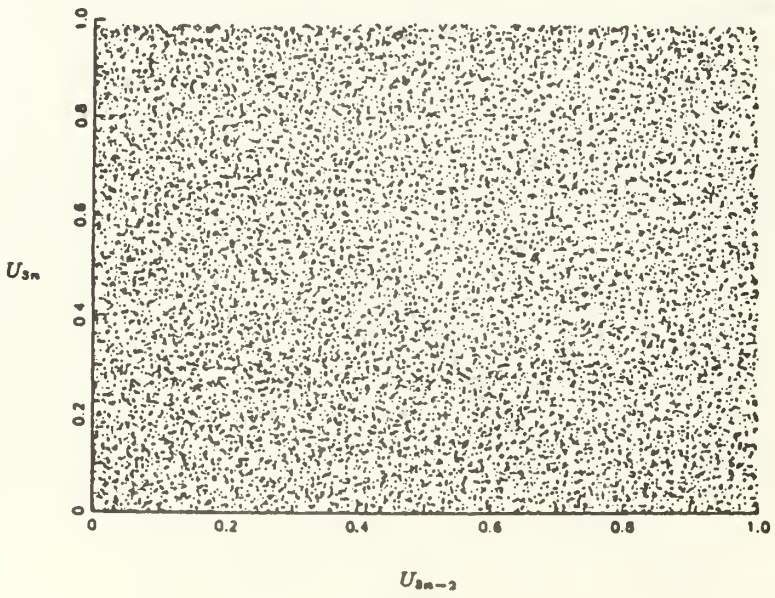

Figure 16: Two-dimensional slice in three-dimensional scatter plot of sequential non-overlapping triples of points from the Wichman-Hill pseudo-random number generator.

DISTRIBUTION LIST

	NO. OF COPIES
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Library (Code 0142) Naval Postgraduate School Monterey, CA 93943-5000	2
Research Administration (Code 012) Naval Postgraduate School Monterey, CA 93943-5000	1
Library (Code 55) Naval Postgraduate School Monterey, CA 93943-5000	1
Center for Naval Analyses 2000 Beauregard Street Alexandria, VA 22311	1
Operations Research Center, Room E40-164 Massachusetts Institute of Technology Attn: R. C. Larson and J. F. Shapiro Cambridge, MA 02139	1
Peter A. W. Lewis (Code 55Lw) Naval Postgraduate School Monterey, CA 93943-5000	322